

Enhancing Medical Database Semantics

Beatriz de F. Leão , MD, PhD(*,**) & Altino Pavan, MSc (**)

(*)Institute of Cardiology RS, Porto Alegre, RS Brazil

(**)Institute of Informatics, Federal University of Rio Grande do Sul, Porto Alegre, RS, Brazil

*Medical Databases deal with dynamic, heterogeneous and fuzzy data. The modeling of such complex domain demands powerful semantic data modeling methodologies. This paper describes **GSM-Explorer** a Case Tool that allows for the creation of relational databases using semantic data modeling techniques. GSM Explorer fully incorporates the **Generic Semantic Data Model - GSM** enabling knowledge engineers to model the application domain with the abstraction mechanisms of generalization/specialization, association and aggregation. The tool generates a structure that implements persistent database-objects through the automatic generation of customized SQL ANSI scripts that sustain the semantics defined in the higher lever. This paper emphasizes the system architecture and the mapping of the semantic model into relational tables. The present status of the project and its further developments are discussed in the Conclusions.*

INTRODUCTION

Semantic data modeling techniques have been proposed as a solution to incorporate more meaning in database design, allowing for a more realistic description of complex domains [1-7]. The first semantic model was published in 1974 [1]. Since then, a major research effort led to the definition of several semantic data models such as the GSM - Generic Semantic Data Model [1-4]. The final goal was to develop powerful mechanisms to describe structural data relationships in a higher abstraction level and translate them into relational database schemes. More recently, database researchers have been investigating the incorporation of dynamic features in these models with strong influence and intersections with the object orientation paradigm [8-12]. Comparing to standard database applications such as Financial or Commercial, Medicine is a rather complex domain to model, where "flat" data-structures are seldom enough to describe the rich semantics of its data. Complex associations are always present such as patients, diagnoses, complications and medical procedures. To fully

represent this complexity one needs powerful semantic data modeling methodologies.

This paper gives an overview of GSM-Explorer - a Case Tool that allows users to model databases using semantic data modeling techniques. This is an ongoing research effort that already lead to one previous SCAMC publication [13] and to one MSc dissertation [14].

GSM EXPLORER ARCHITECTURE

GSM Explorer is an integrated software environment consisting of four layers, as depicts Fig 1.: (1) **GSM Editor & Browser and Consult interface** - a graphical user-interface that incorporates: a modeling tool that fully implements the Generic Semantic Data Model, a Browser facility that shows the object hierarchy and a Consulting interface; (2) **RDBMS interface** - intermediate layer that maps the entities defined in the Editor to its corresponding SQL scripts; (3) **SQL Windows** - external software layer, linked to the application, provides a generic SQL ANSI interface to any RDBMS nowadays used; (4) **SQL Server** - external layer consisting of the RDBMS who holds the databases modeled with the GSM Editor.

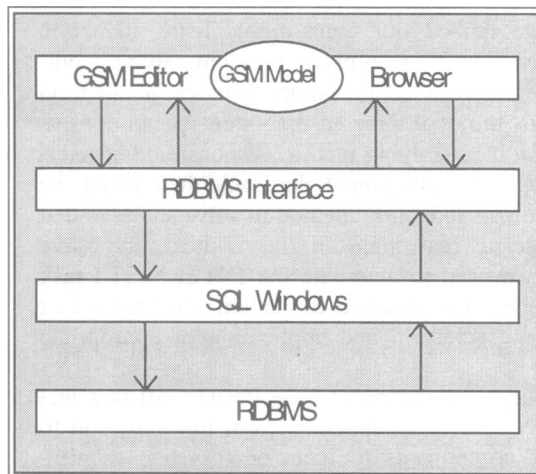


Figure 1 - GSM EXPLORER Architecture

GSM Editor

This is the main component of GSM Explorer users interface. It fully implements the GSM model as defined by HULL in [1]. This is a graphical formalism to model abstract entities. Depending on the abstraction mechanism, the entity has a different graphical representation: type *Abstract entity* (triangle) - represents the highest abstraction level of the system, where the major domain classes are defined; *Subclass* (circle) - children of the abstract entities inheriting all attributes from the parent class, through an *is-a* arch (red-arrow in the graphical user interface); *Association* (*-circle) - implements the abstraction mechanism of association linking the source and destination entities by a *member-of* arch, represented in the model as a combination of *relationship* and *component* arches; *Aggregation* (X-circle) - used to describe the *part-of* abstraction, connecting objects through a *part-of* arch; *attributes* (ellipsis) - printable elements of the model, connected to each entity by a *relationship* (arrow), or a *component* arch, (dotted-arrow).

Figure 2 below shows on the left side the GSM Editor Window where the modeling of two *abstract entities*: *Pts* (patients) and *Diagn* (diagnoses) is depicted. A hierarchy of subclasses is also represented by the circles: *Cardio* and its children: *Rhythm* and *Output*, representing respectively Rhythm and Cardiac Output disorders. Other two subclasses, connected to the root abstract entity are defined in the picture: *Respir* (for respiratory diagnoses) and *Neuron* (for neurological diagnoses). The other *Abstract entity* *Pts* has a printable attribute defined as *name*. These two abstract entities are connected by an association arch

meaning that a *Pts* entity can have associated to it several diagnoses entities. This is expressed in the GSM Editor by a **-arch*, as depicts Figure 2 below. This is indeed one of the most common associations found in any modeling of medical databases, i.e., a Patient usually has several Diagnoses in his/her medical record.

Internally, GSM Explorer represents all graphical elements of the editor as instances of an abstract entity named *GSM_object*. This entity has two sub-classes: *arches* and *polygons*. The *arches* sub-class has the following children: *Is-A-arch*, *relationship* and *component*. The *polygons* sub-class has the following children: *entity*, *sub-class*, *association*, *aggregation* and *attributes*. Specific methods are defined for each children node of *GSM_object*, such as the *connecting* method for the arches objects and the *display* method for all the objects.

Mapping GSM-Editor objects into an object hierarchy.

The mapping of GSM objects into an object hierarchy is straightforward. A generic display method for *Abstract entities* and *sub-classes* objects is evoked, in real-time, displaying the object hierarchy in the object browser window (Fig 2). When the user defines new elements in the Editor window, the corresponding object hierarchy is automatically depicted in the Browser Window. In fact, the data structure used to store the model provides the two views, without any translation effort.

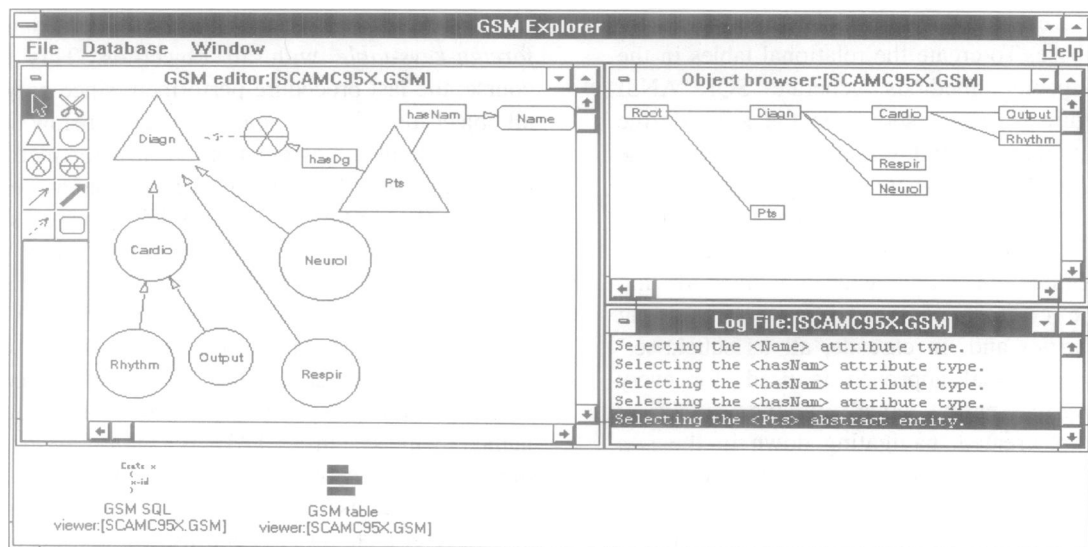


Figure 2. GSM EDITOR and BROWSER

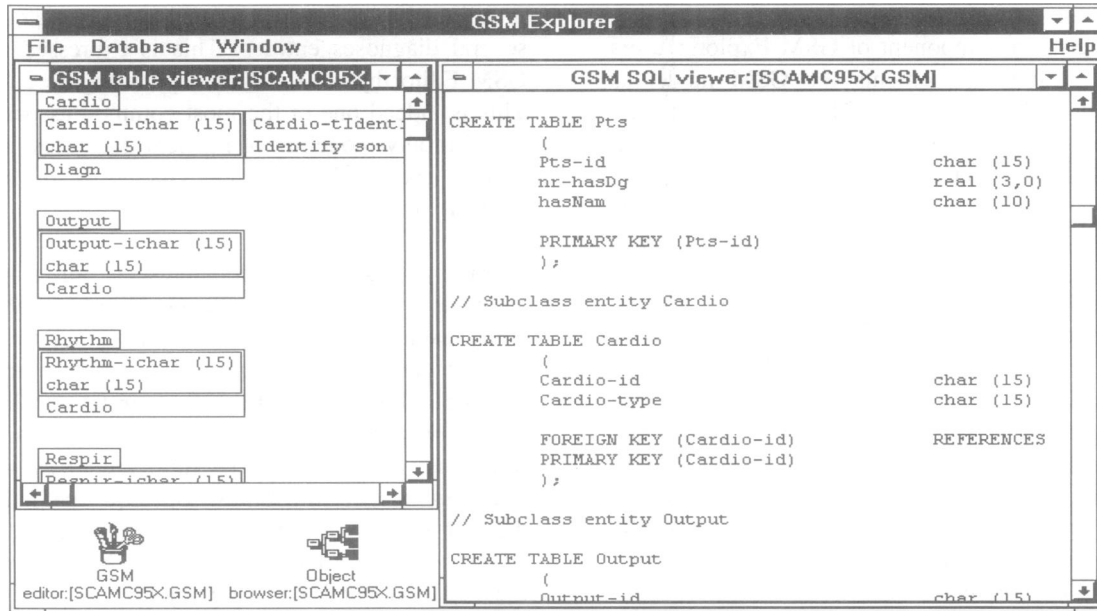


Figure 3. GSM Table Viewer and GSM SQL Viewer

RDBMS interface.

This is the most important layer of GSM- Explorer, where the high level semantics defined in the top layer is mapped to the relational model. This layer, therefore, implements all the necessary procedures to manipulate and create instances defined in the Editor. To start with, while being created, every *abstract entity*, *Subclass* or an *aggregation* receives an *object-id*, automatically generated by GSM-Explorer. The utilization of strong typing and object-ids are an essential feature of this tool. Object persistency is then assured from top to down. Once an object is created in the Editor it becomes persistent in the database. To create the relational tables in the database, specific methods generate SQL ANSI scripts. Figure 3 above shows on the left side the GSM Table Viewer Window, and on the right side the GSM SQL Viewer, where the SQL scripts can be inspected. Note the utilization of the *object-ids* in the definition of the primary and foreign-keys. The database generation starts with a method that triggers the creation of the tables that represent *Abstract entities* and, in cascade, all its subclasses. Therefore, a vertical approach is used: for every *Abstract Entity* (parent node) all the *subclasses* (children) are created, navigating down in the *is-a* hierarchy till the last nodes are found. Each child carries the object-id of its parents. The vertical splitting of the *is-a* hierarchy leads to the creation of several tables with similar attributes, some inherited, some locally created, with the *Parent-object-id* and

Subclass-id as primary keys in each table. Printable elements of the model, ellipsis for GSM-Explorer, are generated according to the standard data-types: Real, Integer, Char and Boolean. Fig 3 above shows, for example, the attribute *hasNam* of the Pts table, defined as type *char*, size 10.

Association Constructor

Whenever the method that is creating the SQL scripts finds an attribute that represents an association, its object-id is stored in a special table named *foreign-keys-table*. After all the abstract entities and its children have been created, the system populates the *foreign-keys-table* with all object-ids. This is, of course, the last procedure performed, since only then all object-ids will be available. Figure 4 below depicts an example of the association constructor method of GSM-Explorer. The Figure depicts the association between the entities: Diagn and PTS. The left side of the Figure 4 shows the GSM Editor Window where the *association-arch*, represented by a **-circle*, connects the two entities. The GSM SQL Viewer Windows is displayed behind, on the left side of the Figure below. It depicts the SQL script to generate this association as a relational table, automatically named by the system as TABLE 334145. To establish the association between these two entities, this table has, therefore, as foreign keys the object-ids of the instances of Diagn and Pts and as primary key a tuple composed by the *object-ids* of the same Diagn and PTS instances.

Model Integrity - the GSM-Editor implements all the integrity constraints of the GSM model. It is, for example, impossible to delete an entity that is being used as foreign key in another instance of the model. Therefore, at the design level, all constraints are guaranteed by the GSM-Editor. The RDBMS interface implements these constraints using foreign-keys that keep track of the *object-id* of the instance it is referring to. Therefore, all arches that represent the relationships of the model, either *is-a*, *member-of* (association) or *part-of* (aggregation) are created using foreign keys and become persistent objects. Since this is a standard procedure in database design, at the physical implementation of the model, its integrity is automatically enforced by the RDBMS. Therefore, the model integrity is preserved from the higher conceptual level to the physical implementation.

The modification of an already existing model with data tables is also possible. All alterations are incremental, preserving the structure already constructed. Specific SQL scripts are then created to modify the structure of an existing table. Whenever necessary, the user is prompted for the final decision, like for instance, when deleting a class that has sub-classes: should all the hierarchy of sub-classes be also

removed? Which attributes of the class that is being removed should be incorporated in the sub-classes?

SQL Windows Layer

The **SQLWindows** layer makes transparent to the user the connection with the RDBMS responsible for storage of all instances of the model. Instead of building a complete module to connect with different RDBMS the choice was to use a software package that already incorporated the great majority of drivers to connect with commercially available RDBMS.

The RDBMS layer

The RDBMS layer is responsible for the storage of the instances created with GSM-Editor. Therefore, the RDBMS allows for the persistence of the objects manipulated by the system.

GSM Browser and Consult Interface

The Browser and Consult interface allows users to navigate and query the database created with GSM-Editor in a graphical user interface. These modules will not be detailed here, however a full description is available in Pavan's MSc dissertation [14].

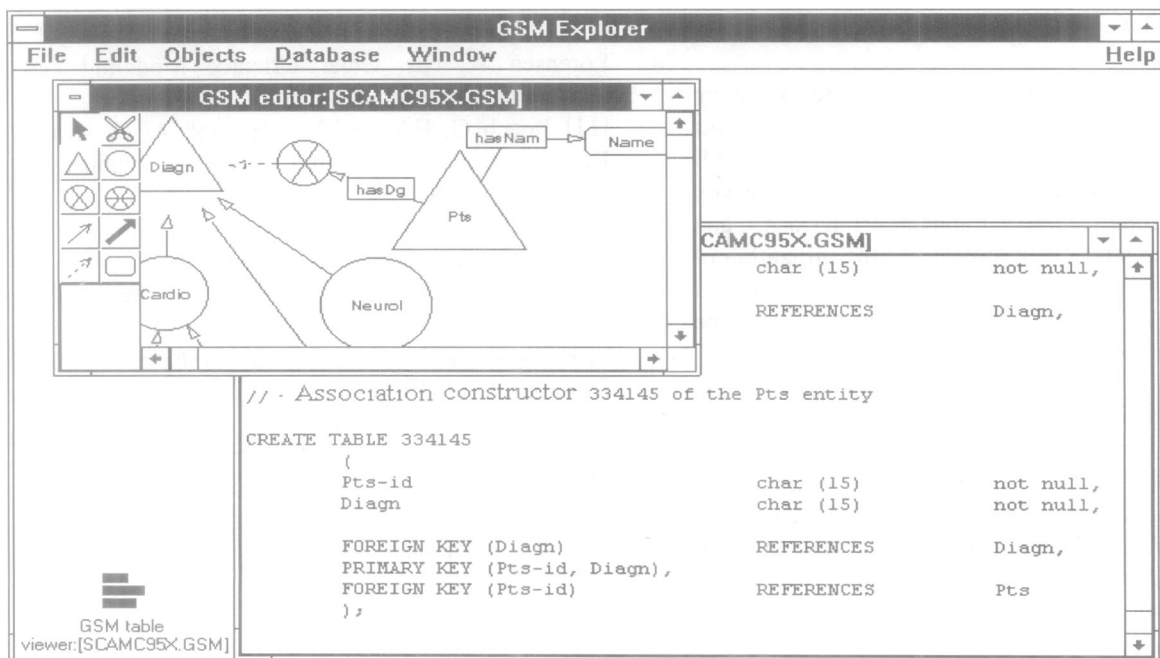


Figure 4. GSM -EXPLORER Association Constructor

CONCLUSIONS

The need for better tools with higher semantics and abstraction mechanisms to model complex domains into a database schema has been challenging the computer science community in the last years[3-7,12-13]. The relational model is by far the most widely used, however poor in semantics. The idea of enhancing the semantics of the relational model by building on top of it an object orientation layer is not new. Several successful attempts have already been reported [8,12]. So far, although very promising, object oriented databases management systems are still not as popular as the relational model. The major problem resides on how to translate the object oriented to the relational model, preserving its integrity. GSM-Explorer follows the approach of enhancing the relational model by building layers of higher semantic on top of it. However, besides the object oriented layer, GSM-Explorer adds an additional functionality offering a more powerful semantic constructor - the GSM-Editor. This allows users to model the application domain not only with *Is-a* hierarchies but also with association and aggregation mechanisms. The use of SQL ANSI standards for the database creation and the utilization of a commercial package to provide connecting facilities with any SQL ANSI server are important features of GSM-Explorer that add robustness and flexibility to the tool.

Beside offering a Case Tool that allows for modeling in a higher abstraction level, for the systems developer the most attractive feature of GSM-Explorer is the object reservoir, that can be easily edited through the graphical interface and incorporated into a new project. As the utilization of this Tool increases, a whole library of medical database objects becomes available and could be shared, even by other institutions.

GSM-Explorer is now being used at the Institute of Cardiology to re-model the Hospital Information System, that runs in a RDBMS. There is still a long way to go before GSM-Explorer becomes a commercial Case Tool, however, so far, it seems to be a very promising approach to model rich domains such as Medicine.

Reference

- [1] Hull R, King R. Semantic Database Modeling: Survey, Applications, and Research Issues. ACM Computing Surveys, 19(3):201-259. September, 1987.
- [2] Abiteboul S, Hull R. A Formal Semantic Database Model. ACM Trans. on Database Systems, 12(4):525-565. December, 1987.
- [3] Codd, E.F. Extending the Database Relational Model to Capture More Meaning. ACM Trans. on Database Systems, 4(4):397-434. December, 1979
- [4] Hammer M, Macleod D. Database Description with SDM: A Semantic Database Model. ACM Trans. on Database Systems, 6(3):351-386. September, 1981.
- [5] Kent W. Limitations of Record-Based Information Models. ACM Trans. on Database Systems, 4(1):107-13. March, 1979.
- [6] Teorey T J, Yang D, Fry J P. A logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model. ACM Comp Surveys, 18(2):197-222. June, 1988
- [7] Abbott R. Knowledge Abstraction. Comm ACM, 30(3):644. August, 1987.
- [8] Rumbaugh, J. Relational Database Design Using an Object-Oriented Methodology. Comm ACM, 31(4):415. April, 1988.
- [9] Stroustrup B. What is Object-Oriented Programming? IEEE Software, 5(3):10. May, 1988.
- [10] Rumbaugh J, Blaha M, Premerlani W, Eddy F, Lorensen W. eds. Object-Oriented Modeling and design, Prentice Hall, New Jersey, 1991.
- [11] Booch G. Edt. Object-Oriented Analysis and Design, The Benjamin/Cummings Publishing Company, Santa Clara, 1994.
- [12] Wiederhold G. Views, Objects and Databases. IEEEComputer, 19 (12) 377:44, December 1986.
- [13] Leao BdF, Mantovani R, Rossi RI., Zielinsky P. Incorporating knowledge to databases - a solution to complex domains. Proc. of the Sixteenth Annual Symposium on Computer Applications in Medical Care, 234-238, November, 1992.
- [14] Pavan A. Uma Ferramenta Case para Modelagem Semântica de Bancos de Dados Relacionais. Msc Dissertation. Institute of Informatics. UFRGS. Porto Alegre, 1995.